沿用原本系统内约定俗称的方式实现的功能我就不写了，其他同志想要接手应该不存在理解障碍。这里就写两个和之前的做法不太一样的东西：

1. Common/Utilities
2. Logic/GuideManager

这个文档请配合代码阅读。

Utilities 的用法可以在 UIMinimapArea 里面找到用法的例子。Utilities 是用来减少前端的 GetChild(XXX).asXXX 之类的重复的，因为 FairyGUI 里画出界面后，大量的组件需要以这种方式在 Lua 脚本里使用，这些 GetChild(XXX).asXXX 的代码在 Lua 各处重复调用，所有组件的状态、行为都没有一个有序的组织。所以我就写了这么个 Utilities 对象。

Utilities 主要思路就是，一个对象来描述 FairyGUI 里画出的界面，这个对象的所有字段直接对应到界面的一个组件，字段名称就是 FairyGUI 里给出的组件名称，字段值也是个小对象，这个对象有一个必填字段是 comp\_type 说明了对应的组件的类型，两个可选字段是 cb\_init 和 cb\_update，这两个字段都是回调方法，接受组件对象作为参数。cb\_init 在调用 Utilities.init\_ui 的时候会被调用，预期它应当在界面初始化的时候被调用；cb\_update 在调用 Utilities.update\_ui 的时候被调用，预期它在界面需要被刷新的时候使用。

我在写 Lua 脚本的时候，所有 cb\_ 开头的变量都是指它是个回调方法（cb 是 callback 的缩略）。

GuideManager 是新手引导的所有逻辑的集中地，预期所有的新手引导相关的逻辑应当都在这个脚本中。脚本中有个局部变量叫 script，是所有新手引导逻辑的容器，本身是个数组，每一个数组元素都对应到新手引导文档中的一个步骤。数组元素是一个对象，章节的概念是通过其 head 字段来实现的，head = true 的元素标识着新章节的开始；save = true 的字段标识着章节的存储点。按新手引导的需求，head、save 字段在一章中，都分别只能有一个数组元素具备。数组元素的另外两个字段是 name 和 exec，前者是章节名称，一个是为了脚本开发者可读，一个是为了将来数据采集提供一点信息；后者是一个方法，接受一个回调方法作为参数。exec 方法就是当前引导步骤的行为，里面只要保证它接收的回调方法一定会被调用，就能保证引导的流程继续。

GuideManager 的 script 里的大多数步骤可能都需要延时一段时间才能进入下一步，这个时候有个常用的实现套路是设置回调方法，比如需要在某个界面上高亮引导某个按钮点击，实际上引导走到这一步时，目标界面还没显示出来，目标按钮当然也找不到影子，这个时候必须要保证设置高亮的动作必须发生在界面显示了、按钮创建了之后，就需要在目标界面的代码里放一个后门，我之前实现的时候，这个后门回调一致命名为 cb\_backdoor，用这个后门的做法，可以将新手引导对目标界面的影响降到最低。在目标界面的逻辑的合适的地方调用这个 cb\_backdoor，就可以保证高亮引导显示在正确的位置。具体可以参考 UIOfficerList。